Seminar 2,

Logical and Physical Model

Data Storage Paradigms, IV1351

Esra Salman

esras@ug.kth.se

22/11 – 2023

Table of Contents

1 Introduction

Seminar 2 is the second phase in developing a single database application. The objective of this project is to create a database for the *Soundgood music school co* for information handling and business transaction facilitation, including all data and several specified operations. The foundation of the school is built on selling music lessons to students, which are offered in various categories and levels. The intended learning outcomes are stated as follows: *Model needs for information based on an organizational description and convert the model to a functioning database* and usage of *relational databases and query languages* (Canvas, *IV1251*).

With the objective to practice logical and physical models, relational model and algebra, normalization, and SQL, the preparation of this seminar included reading chapters 5, 8, 9, 14 and 15 in the provided course literature *Fundamentals of Database Systems* alongside the lectures in *Logical and Physical Models, The Relational Model + Relational Algebra* and *SQL – The Structured Query Language* modules in canvas. The subject's relational databases, set theory, schema, relation, valid database state, integrity constraints, and so on.

The main objective of this seminar is to translate the *Conceptual Model* from previous task to a *Logical* & *Physical Model* with physical aspects, covering the entirety of the school's provided description. A Logical Model defines what to store in the data storage, i.e., relational database, and does not handle views or other physical storages. The Physical Model specifies how the data in the Logical Model is stored, which is also adapted to the database type. In this task one model is creates with both prerequisites for Logical and Physical Model.

This seminar task was made alongside Daniel Ibrahimi and Ermia Ghaffari

2 Method

To start this seminar, an extensive preparation was made by thoroughly examining ch. 5, 8-9, 14-15 in the course literature, "Fundamentals of Database Systems" as well as a comprehensive review of lectures *Logical and Physical Models, The Relational Model + Relational Algebra* and *SQL – The Structured Query Language*.

As mentioned above, seminar 2 primarily consists of designing a Logical and Physical Model. In accordance with the requirements, decisions were made in modeling in regards to the data provided. The UML modeling application *Astah professional* and the open-source database management system *PostgreSQL* (in the database management tool *pgAdmin*) was used for the given tasks. Initially began by brainstorming possible relations, then doublechecking the relations' normalized states, followed by confirming the operations specified in the model are performable, and finally attribute types were defined. Moreover, the Physical Model approach started with adapting the DBMS, storage details, and security. Since the model created is merging both the Logical and Physical model conditions, both were accounted for.

First the tables were created in *Astah* using the Conceptual Model as a foundation, followed by inserting the attributes with at most one value, then filling the tables for a higher cardinality and specifying the domain for each. Consequently, column constraints *unique* and *not null* were applied to relevant attributes, followed by assigning primary keys to entities. The lecturer mentioned the preference of surrogate keys as a default resulting in avoidance of duplicates in *Logical and Physical Models, Part 2*.

Following the assigning of primary keys, relations *one-to-one* and *one-to-many* are drawn, alongside foreign keys assigned on the weak end of the relation entity with the related primary key. Then foreign key constraints are considered, with the three actions: prohibit (NO ACTION), allow (SET NULL), and delete (CASCADE).

Subsequent to considering constraint for *one-to-one* or *one-to-many* relations, *many-to-many* are processed. By using existing primary keys as foreign keys and considering the relevancy of primary keys to be foreign keys in combination or a surrogate one, then all constraints. The last step in the process of building a Logical and Physical Model from the Conceptual, is assigning foreign keys to the entities with multi-valued attributes. The above steps were iterated multiple times until all tables were complete and the model fulfilled the requirements.

Afterwards, the model is assessed for normalization which is the process of organizing and structuring the data to achieve less redundancy and dependency. In a Logical model this entails defining relations between tables and reducing data redundancy, whereas the Physical model focuses on optimization of performance and storage, with the objective to improve query performance.

Finally, when the model is complete all queries are stapled in relevance to their hierarchical relation. Consequently, data is generated through a data generator tool (https://generatedata.com/) and tested through the data management application.

3 Result

The Logical & Physical Model and queries published in GitHub:

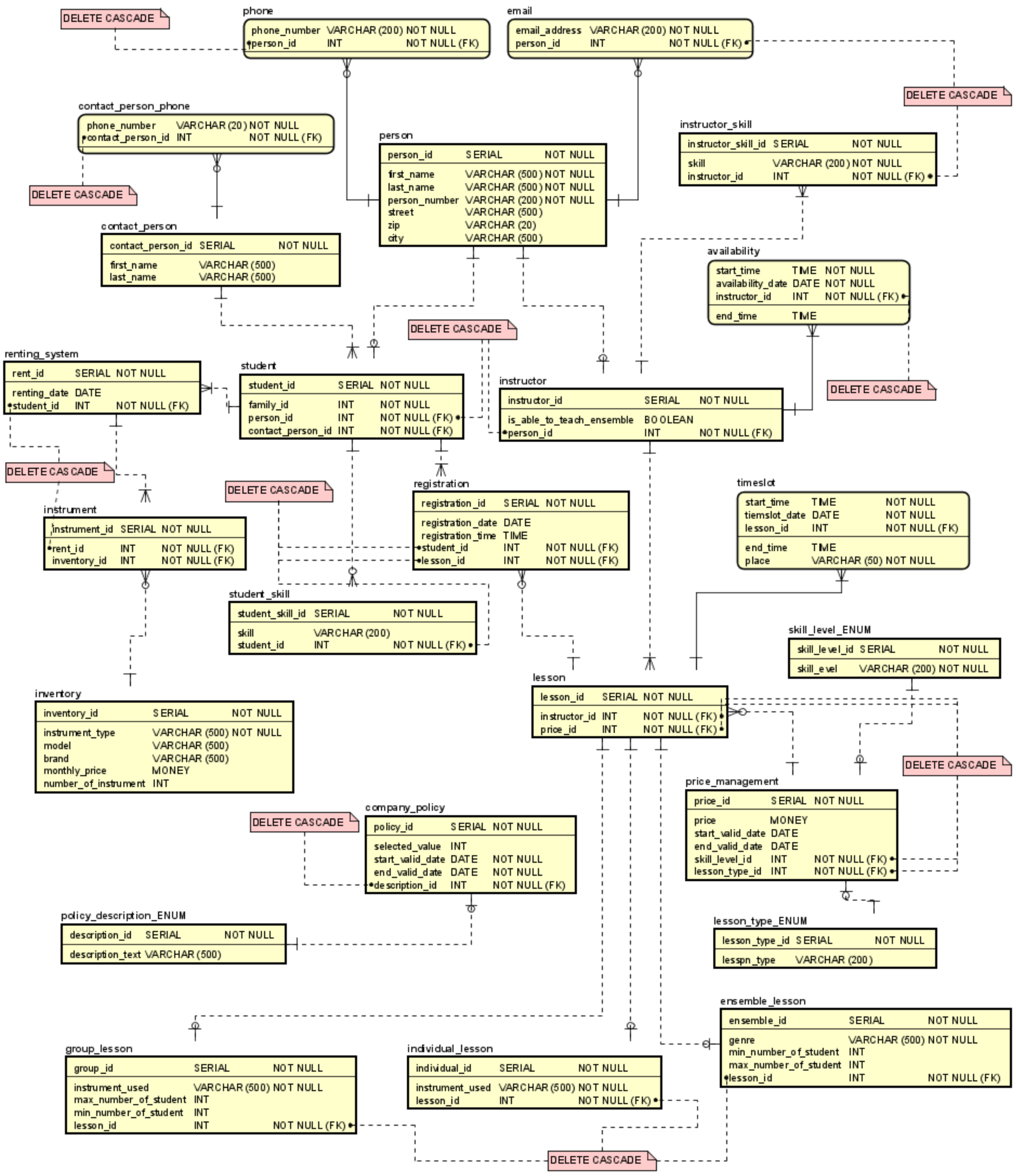
https://github.com/esrsal/IV1351.git

Logical & Physical Model:

In fig 3.1 a visual representation of the model with all tables, cardinalities, relations, keys, and data types is shown. Similar to the previous model, the Logical & Physical Model contains all previous tables with a few newly added. Some inheritance relations are still intact, for instance *student* and *instructor* still inherit attributes from *person.* Additionally, the primary key *person_id* is iterated in both *student* and *instructor* as foreign keys. Although the *person_id* has a SERIAL in *person* it is changed in the other two as INT, since the consecutive order is not relevant in the child entities.

Moreso, *lesson* and t*imeslot* have the attribute *lesson_id* and in the inheriting table *timeslot* the attribute is both a foreign key and primary key to avoid any duplicates, i.e., if a lesson with a specific *lesson_id* is booked it cannot be booked again with the same timeslot. Furthermore, the table *policy* is an umbrella term for various requirements, for instance the number of instruments allowed to be rented by students, dates for the renting, alongside the description for the kind of policy, in this case instrument renting. Another policy considered is the length of renting, as the school requires a limit for 12-month period. Also, the discount is also included as student could be eligible to a reduced price. This is possible by *student_id* attribute listed on these tables, withdrawing the information needed from *renting_system*.

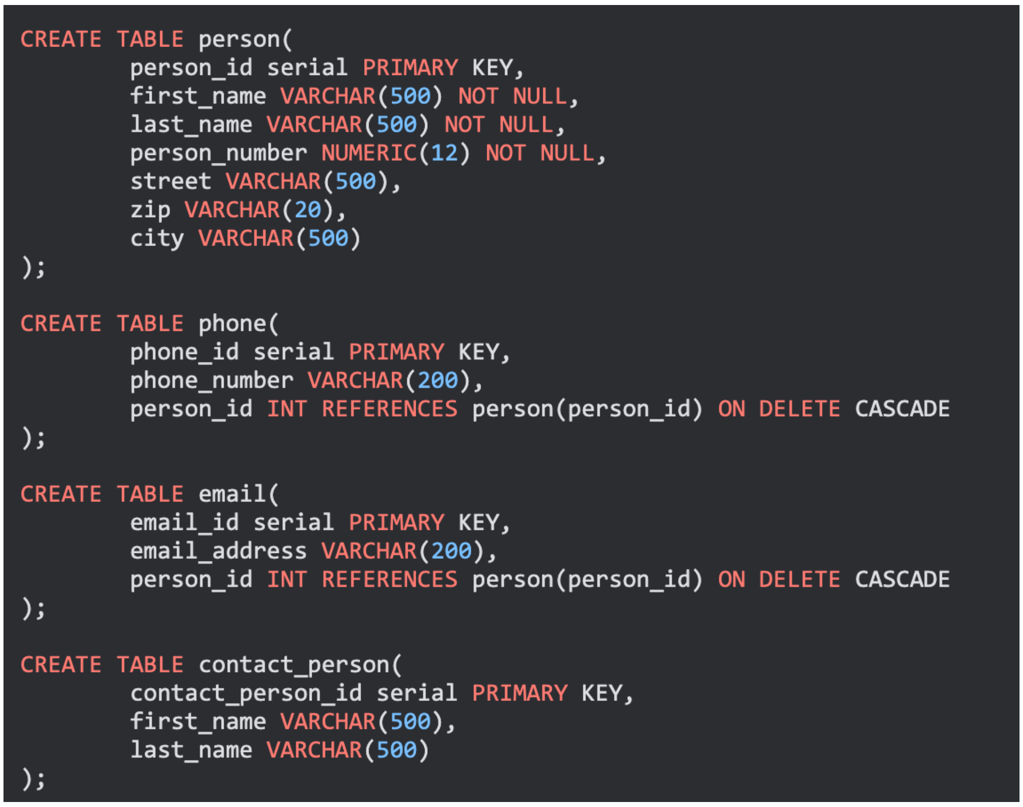
Regarding the *policy* table another ENUM table is connected, *policy_description_ENUM* that contains the attribute *description* as a VARCHAR enabling it to contain a long description of the *description_id.* Notwithstanding that ENUM, *Electronic Numbering*, limits the options available for the user when selecting, i.e., limiting the domain. Another instance of ENUM is *lesson* and *lesson_type_ENUM* where a user can select a lesson type (beginner, intermediate, advanced) as a predetermined options rather than typing the skill level. Accordingly, *price_management* which is also cardinal to *lesson* results in a few outcomes depending on the dates the student is booking a lesson. Since a timeframe, skill level and lesson type are mentioned as attributes the price is adjusted depending on the date, among other things.



DELETE CASCADE
phone
phone_number VARCHAR (200) NOT NULL
person_id INT NOT NULL (FK)
email
email_address VARCHAR (200) NOT NULL
person_id INT NOT NULL (FK)
contact_person_phone
phone_number VARCHAR (20) NOT NULL
contact_person_id INT NOT NULL (FK)
person
person_id SERIAL NOT NULL
first_name VARCHAR (500) NOT NULL
last_name VARCHAR (500) NOT NULL
person_number VARCHAR (200) NOT NULL
street VARCHAR (500)
zip VARCHAR (20)
city VARCHAR (500)
instructor_skill
instructor_skill_id SERIAL NOT NULL
skill VARCHAR (200) NOT NULL
instructor_id INT NOT NULL (FK)
contact_person
contact_person_id SERIAL NOT NULL
first_name VARCHAR (500)
last_name VARCHAR (500)
availability
start_time TIME NOT NULL
availability_date DATE NOT NULL
instructor_id INT NOT NULL (FK)
end_time TIME
renting_system
rent_id SERIAL NOT NULL
renting_date DATE
student_id INT NOT NULL (FK)
student
student_id SERIAL NOT NULL
family_id INT NOT NULL
person_id INT NOT NULL (FK)
contact_person_id INT NOT NULL (FK)
instructor
instructor_id SERIAL NOT NULL
is_able_to_teach_ensemble BOOLEAN
person_id INT NOT NULL (FK)
instrument
instrument_id SERIAL NOT NULL
rent_id INT NOT NULL (FK)
inventory_id INT NOT NULL (FK)
registration
registration_id SERIAL NOT NULL
registration_date DATE
registration_time TIME
student_id INT NOT NULL (FK)
lesson_id INT NOT NULL (FK)
timeslot
start_time TIME NOT NULL
tiemslot_date DATE NOT NULL
lesson_id INT NOT NULL (FK)
end_time TIME
place VARCHAR (50) NOT NULL
student_skill
student_skill_id SERIAL NOT NULL
skill VARCHAR (200)
student_id INT NOT NULL (FK)
skill_level_ENUM
skill_level_id SERIAL NOT NULL
skill_evel VARCHAR (200) NOT NULL
inventory
inventory_id SERIAL NOT NULL
instrument_type VARCHAR (500) NOT NULL
model VARCHAR (500)
brand VARCHAR (500)
monthly_price MONEY
number_of_instrument INT
lesson
lesson_id SERIAL NOT NULL
instructor_id INT NOT NULL (FK)
price_id INT NOT NULL (FK)
price_management
price_id SERIAL NOT NULL
price MONEY
start_valid_date DATE
end_valid_date DATE
skill_level_id INT NOT NULL (FK)
lesson_type_id INT NOT NULL (FK)
company_policy
policy_id SERIAL NOT NULL
selected_value INT
start_valid_date DATE NOT NULL
end_valid_date DATE NOT NULL
description_id INT NOT NULL (FK)
policy_description_ENUM
description_id SERIAL NOT NULL
description_text VARCHAR (500)
lesson_type_ENUM
lesson_type_id SERIAL NOT NULL
lesspn_type VARCHAR (200)
ensemble_lesson
ensemble_id SERIAL NOT NULL
genre VARCHAR (500) NOT NULL
min_number_of_student INT
max_number_of_student INT
lesson_id INT NOT NULL (FK)
group_lesson
group_id SERIAL NOT NULL
instrument_used VARCHAR (500) NOT NULL
max_number_of_student INT
min_number_of_student INT
lesson_id INT NOT NULL (FK)
individual_lesson
individual_id SERIAL NOT NULL
instrument_used VARCHAR (500) NOT NULL
lesson_id INT NOT NULL (FK)

Figure 3.1: Logical and Physical model with relevant tables, attributes, relations, and keys.

Queries for database:



```sql
CREATE TABLE person(
        person_id serial PRIMARY KEY,
        first_name VARCHAR(500) NOT NULL,
        last_name VARCHAR(500) NOT NULL,
        person_number NUMERIC(12) NOT NULL,
        street VARCHAR(500),
        zip VARCHAR(20),
        city VARCHAR(500)
);

CREATE TABLE phone(
        phone_id serial PRIMARY KEY,
        phone_number VARCHAR(200),
        person_id INT REFERENCES person(person_id) ON DELETE CASCADE
);

CREATE TABLE email(
        email_id serial PRIMARY KEY,
        email_address VARCHAR(200),
        person_id INT REFERENCES person(person_id) ON DELETE CASCADE
);

CREATE TABLE contact_person(
        contact_person_id serial PRIMARY KEY,
        first_name VARCHAR(500),
        last_name VARCHAR(500)
);
```

Figure 3.2: Queries for some tables in relevant hierarchical order.

In fig 3.2 an excerpt of the queries is shown. The table *person* is higher up due to its function as an inheritor, or parent entity, with inheriting entities lower on the list.

4 Discussion

The Logical & Physical Model presented in this seminar is designed with all requirements taken into consideration, as mentioned in section 2. In the requirements description the type of roles, objects, and systems are provided, all of which are included as tables in the model. Moreover, attributes and relationships are drawn, discussed, reevaluated, and confirmed with the teacher, to ensure their correction. Further, the aspects of interest are the policy of the school, the use of ENUMS in the model, and the *price_managment* table. In this section the reasoning behind these aspects is thoroughly discussed and analyzed.

In section 3 *Result,* the *policy* table is representing the school's policy for various proceedings. Some of these include the renting limit for students, which in this case is two instruments per student. To enable the user to know the number of instruments burrowed by the students, the *renting_system* is utilized since *student_ID* is present in the table. The user can easily count by searching through the record data, through this process an amount is retrieved and then compared to the policy, checking whether the student is allowed to rent an instrument/-s. As shown in fig. 3.1 *policy* table is related to *policy_description_ENUM,* which limits the domain from including variations in the data provided.

ENUM, as explained in section 2 and 3, is the restricting of the domain in the relevant table. There are several benefits with ENUMs, to follow though the *policy_description_ENUM*, If the description involves a lengthy string detailing a specific policy, it may necessitate hundreds of characters for a comprehensive articulation. In scenarios where the company intends to update the policy value while retaining the previous record in the system, there is no requirement to transfer the entire description to the new entry in the policy table. Instead, a new *policy_ID* and *value* can be assigned, still referencing the same description using an integer. This approach leads to substantial storage savings in the database and eliminates the need for unnecessary duplicates. This is the objective of using ENUMs, to simply maintain consistency throughout the database.

Finally, since this task's objective was to devise a model that accommodates the dynamic nature of lesson prices another entity was required, *price_management*. Given the possibility of price fluctuations, particularly if a student attended a lesson a previous month with subsequent changes in pricing, the system needed to alter details for the lesson price prevailing at the time of the student's attendance. The model crafted in this experiment tackles this challenge by introducing entities dedicated to *price_management*, registration, and lessons. For instance, when a student schedules a lesson, they undergo registration in the system through a registration entity.This recorded information can then be leveraged to conclude the correct price at time of registration.